

JEE: formulaires

Achref El Mouelhi

Docteur de l'université d'Aix-Marseille
Chercheur en programmation par contrainte (IA)
Ingénieur en génie logiciel

elmouelhi.achref@gmail.com



Un formulaire

- un outil graphique
 - que nous créons avec le langage de description **HTML**
 - que nous gérons avec un langage de programmation tel que **PHP**, **Java**...
- il permet à l'utilisateur de saisir des données
- et de les envoyer vers une autre page, vers une base de données...

JEE : formulaires

Contenu de la page ajoutPersonne.jsp

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Ajout</title>
  </head>
  <body>
    <form method="POST" action="ajoutPersonne">
      <div>Formulaire d'ajout d'une Personne</div>
      <div><label for="nom">Nom *</label>
        <input type="text" id="nom" name="nom" value="" >
      </div>
      <div><label for="prenom">Prénom *</label>
        <input type="text" id="prenom" name="prenom" value="" >
      </div>
      <input type="submit" value="Ajouter">
    </form>
  </body>
</html>
```

Déclaration d'un formulaire :

```
<form method="POST ou GET" action="url-pattern">  
...  
</form>
```

JEE : formulaires

Déclaration d'un formulaire :

```
<form method="POST ou GET" action="url-pattern">  
    ...  
</form>
```

Les attributs d'un formulaire

- **action** : indique l'`url-pattern` d'une servlet (tel qu'on l'a déclarée dans le `web.xml` ou avec l'annotation `@WebServlet(...)`)
- **method** : concerne l'envoi de données et peut prendre deux valeurs.
 - `GET` : c'est la méthode `doGet()` de la servlet qui sera appelée.
 - `POST` : c'est la méthode `doPost()` de la servlet qui sera appelée.

Par rapport à l'exemple précédent

- Quand on saisit l'url `.../ajoutPersonne` dans le navigateur, c'est la méthode `doGet()` qui sera appelée.
- Quand on valide le formulaire, il est préférable d'appeler la méthode `doPost()`.

JEE : formulaires

Les doGet() et doPost() de AjoutPersonneServlet annotée par
@WebServlet("/ajoutPersonne")

```
protected void doGet(HttpServletRequest request, HttpServletResponse
    response) throws ServletException, IOException {
    this.getServletContext().getRequestDispatcher("/WEB-INF/ajoutPersonne
        .jsp").forward(request, response);
}

protected void doPost(HttpServletRequest request, HttpServletResponse
    response) throws ServletException, IOException {
    String nom = request.getParameter("nom");
    String prenom = request.getParameter("prenom");
    Personne personne = new Personne(nom, prenom);
    PersonneDaoImpl daop = new PersonneDaoImpl();
    Personne insertedPersonne = daop.save(personne);
    request.setAttribute("personne", insertedPersonne);
    request.getRequestDispatcher("/WEB-INF/confirmation.jsp").forward(
        request, response);
}
```

JEE : formulaires

Contenu de confirmation.jsp

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Confirmation</title>
  </head>
  <body>
    <h2>Page de confirmation</h2>
    <c:out value="La personne nommée ${ personne.
      prenom } ${ personne.nom } a bien été ajoutée
      dans la base de données.">
    </c:out>
  </body>
</html>
```

Contraintes sur les champs du formulaire précédent

- Interdire l'insertion vide (pour les deux champs nom et prénom).
- Pas de chaîne de longueur inférieure à deux.
- Première lettre en majuscule.

JEE : formulaires

Contraintes sur les champs du formulaire précédent

- Interdire l'insertion vide (pour les deux champs nom et prénom).
- Pas de chaîne de longueur inférieure à deux.
- Première lettre en majuscule.

Démarche à faire en cas de contrainte non-respectée

- Re-afficher le même formulaire (`ajoutPersonne.jsp`).
- Afficher un message d'erreur.
- Garder les valeurs saisies affichées dans le formulaire.

JEE : formulaires

Dans AjoutPersonneServlet, ajoutons la méthode verifChaine() qui true si les contraintes précédents sont respectées, false sinon.

```
public boolean verifChaine(String s) {  
    if (s == null || s.length() < 2) {  
        return false;  
    }  
    char c = s.charAt(0);  
    if (!(c >= 'A' && c <= 'Z')) {  
        return false;  
    }  
    for(int i = 0; i < s.length(); i++) {  
        c = s.charAt(i);  
        if (!(c >= 'a' && c <= 'z')&& !(c >= 'A' && c <= 'Z')) {  
            return false;  
        }  
    }  
    return true;  
}
```

JEE : formulaires

Préparons le formulaire pour afficher les messages d'erreurs et les valeurs saisies par l'utilisateur en cas d'erreur

```
<form method="POST" action="ajoutPersonne">
    <div>Formulaire d'ajout d'une Personne</div>
    <div>
        <label for="nom">Nom *</label>
        <input type="text" id="nom" name="nom" value="${nomSaisi}">
        ${ nomIncorrect }
    </div>
    <div><label for="prenom">Prénom *</label>
        <input type="text" id="prenom" name="prenom" value="${prenomSaisi}">
        ${ prenomIncorrect }
    </div>
    <input type="submit" value="Ajouter">
</form>
```

Modifions doPost() dans AjoutPersonneServlet

```
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    String nom = request.getParameter("nom");
    String prenom = request.getParameter("prenom");
    boolean verifNom = verifChaine(nom);
    boolean verifPrenom = verifChaine(prenom);
    if (!verifNom)
        request.setAttribute("nomIncorrect", "format incorrect");
    if (!verifPrenom)
        request.setAttribute("prenomIncorrect", "format incorrect");

    if (!verifNom || !verifPrenom){
        request.setAttribute("nomSaisi", nom);
        request.setAttribute("prenomSaisi", prenom);
        request.getRequestDispatcher("/WEB-INF/ajoutPersonne.jsp").forward(
            request, response);
    }
    else {
        // utiliser le dao pour sauvegarder la personne
        request.getRequestDispatcher("/WEB-INF/confirmation.jsp").forward(
            request, response);
    }
}
```

Une deuxième solution

Les exceptions

© Achref EL MOUADJI

Une deuxième solution

Les exceptions

Remarque

Aucune modification à apporter aux vues.

JEE : formulaires

Utilisons les exceptions dans verifChaine()

```
public void verifChaine(String s) throws Exception {
    if (s == null || s.length() < 2) {
        throw new Exception("La chaîne doit comporter au moins deux
                            caractères");
    }
    char c = s.charAt(0);
    if (!(c >= 'A' && c <= 'Z')) {
        throw new Exception("La chaîne doit commencer par une
                            lettre en majuscule");
    }
    for(int i = 0; i < s.length(); i++) {
        c = s.charAt(i);
        if (!(c >= 'a' && c <= 'z') && !(c >= 'A' && c <= 'Z')) {
            throw new Exception("La chaîne ne peut contenir que des
                                lettres");
        }
    }
}
```

```
protected void doPost(HttpServletRequest request, HttpServletResponse
    response) throws ServletException, IOException {
    boolean test = true;
    String nom = request.getParameter("nom");
    String prenom = request.getParameter("prenom");
    try {
        verifChaine(nom);
    } catch (Exception e) {
        request.setAttribute("nomIncorrect", e.getMessage());
        test = false;
    }
    // refaire la même chose pour prenom
    if (test){
        // utiliser le dao pour sauvegarder la personne
        request.getRequestDispatcher("/WEB-INF/confirmation.jsp").forward(
            request, response);
    }
    else {
        request.setAttribute("nomSaisi", nom );
        request.setAttribute("prenomSaisi", prenom);
        request.getRequestDispatcher("/WEB-INF/ajoutPersonne.jsp").forward(
            request, response);
    }
}
```

Exercice 1

- Créez une Servlet `ModifPersonneServlet` accessible via la route `/modifPersonne`.
- La méthode `doGet()` de `ModifPersonneServlet` affiche une vue `modifPersonne.jsp` (contenant un formulaire avec trois champs : un premier pour le num, un deuxième pour le nom et un dernier pour le prenom).
- La méthode `doPost()` de `ModifPersonneServlet` récupère les données de `modifPersonne.jsp`, modifie la personne selon l'identifiant dans le service (ou la base de données) et affiche la vue confirmation.

Exercice 2

- Créez une **Servlet** SuppPersonneServlet accessible via la route /suppPersonne.
- La méthode `doGet()` de SuppPersonneServlet affiche une **vue** suppPersonne.jsp (contenant un formulaire avec un seul champs : le num).
- La méthode `doPost()` de SuppPersonneServlet récupère les données de suppPersonne.jsp, modifie la personne selon l'identifiant dans le service (ou la base de données) et affiche la **vue** confirmation.

Exercice 3

- Modifiez la vue confirmation.jsp et adaptez son message à l'opération réalisée. Par exemple, pour la suppression, on affiche La personne nommée ... a bien été supprimée de la base de données.
- On veut aussi afficher la liste des personnes juste après le message précédent.
- Créez une Servlet ConsultPersonneServlet avec une méthode doGet() qui affiche la vue confirmation.jsp sans le message de confirmation (Que la liste des personnes).

Exercice 4

- Nous voudrons fusionner les trois Servlets AjoutPersonneServlet, ModifPersonneServlet et SuppPersonneServlet en une seule Servlet PersonneServlet et les trois vues ajoutPersonne.jsp, modifPersonne.jsp et suppPersonne.jsp en une seule personne.jsp.
- PersonneServlet est accessible via trois routes addPerson, editPerson et removePerson et retourne chaque fois la même vue : personne.jsp.
- Le formulaire à afficher dans personne.jsp dépend de la route demandée. Par exemple, si la route saisie est addPerson, on affiche le titre Ajouter Personne avec un formulaire contenant deux champs : un pour le nom et un pour le prénom et un bouton avec l'étiquette Ajouter.
- Les données du formulaire de personne.jsp seront envoyé à la méthode doPost de PersonneServlet qui réalise l'opération demandée et affiche la vue confirmation.